Symbolic Algebra and Renormalization of Gauge Theories

M. Rossi and A.P. Flitney

*Department of Physics and Mathematical Physics, University of Adelaide,
Adelaide, South Australia 5005, Australia*

July 24, 1995

Abstract

Symbolic algebra relevant to the renormalization of gauge theories can be efficiently performed by machine using modern packages. We devise a scheme for representing and manipulating the objects involved in perturbative calculations of gauge theories. This scheme is readily implemented using the general purpose package, Mathematica. The techniques discussed are used to calculate renormalization group functions for a non-abelian $SU(m)$ gauge theory with massless fermions in a representation R, in the two-loop approximation, and to simplify some expressions arising in electroweak calculations at the two loop level.

Correspondence to A.P. Flitney at the address given
E-mail: *aflitney@physics.adelaide.edu.au*

1 Introduction

Perturbation theory has been an important calculational method in quantum field theory for several decades. However the practical usefulness of perturbation theory is limited mainly by the rapid rise in the amount of labour required to improve the order of approximation. For instance, the renormalization group functions can be calculated analytically to three loops in non-abelian gauge theory, which requires the evaluation of 440 three loop diagrams, and to five loops in $\lambda\phi^4$ theory [1,2]. The non-numerical nature of such perturbative calculations in quantum field theory has complicated the effort to automate these calculations. However, the development of specialised computer algebra systems [3] and improvements in general purpose computer algebra systems have greatly facilitated this task [4–9].

In devising a scheme for the perturbative calculation of amplitudes in quantum field theory the main areas which need to be developed are:

- The perturbation expansion itself,

- Lorentz tensor, Dirac and symmetry group algebra, and

- Feynman integration.

The perturbation expansion is either a Dyson-Wick expansion in the canonical formalism or equivalently an expansion of the generating functional in the path integral formalism. The perturbation expansion will not concern us here, nor will we be concerned with automating symmetry group algebra which is simple for all cases of interest to us. Our focus will be on Lorentz tensor and Dirac algebra, and on Feynman integration.

There are a number of existing packages that tackle the problem of Dirac algebra computations [4–6,8]. The symbolic evaluation of Feynman diagrams at tree level [6] and one loop [9] have been approached using Wolfram's computer algebra package Mathematica [10]. Fleischer and Tarasov [7] present a package for the evaluation of certain two loop Feynman integrals written in the computer algebra language FORM [11], while symbolic three loop calculations [12] have also been carried using FORM. The current paper presents a package that combines a method for simplifying the Dirac algebra with procedures for evaluating massless scalar or tensor Feynman integrals at one or two loops, and is implemented in Mathematica.

The following sections discuss our scheme for representing Feynman diagrams. The scheme is essentially a definition of a notation that can be readily expressed in Mathematica. Section two introduces the notational scheme, section three will discuss some aspects of rules which have been encoded to simplify and integrate expressions, and section four demonstrates the evaluation of some specific diagrams. Usage messages for functions defined by us can be found in the appendix.

2 Notational Scheme

In this section it will be shown how one can use the symbolic algebra capabilities of Mathematica to represent and simplify the various types of tensorial and matrix expressions that arise in amplitudes of gauge theories. This is based upon a suitable notation for dimensionally continued tensors and Dirac gamma matrices. The aim is to establish a notation that can be readily accommodated by the symbolic capabilities of Mathematica. Dirac algebra can be performed without reference to any representation of the Dirac matrices, by using the commutation relations which define the algebra and identities derived from those relations, along with the rules of matrix multiplication. Tensor manipulations are also readily implemented symbolically. As we are interested in evaluating amputated one particle irreducible amplitudes which have been regularised using dimensional regularisation [13], no explicit representation of Dirac gamma matrices is necessary.

The first step is to separate symbols which will represent four-vectors from symbols which will represent tensor indices. We generally use one or more lower case letters and numerals which are not otherwise defined, such as a,k,p1 ... for four-vectors and indices. The two types of symbols can be distinguished by declaring a list of symbols which will represent four-vectors. Any symbols which appear that are not in this list and are not otherwise defined shall be understood to represent a Lorentz index, and need not be declared. These symbols will appear only as the arguments of Mathematica expressions. The expressions themselves will represent Dirac matrices, Lorentz scalars, four-vectors and tensors depending on the symbols that appear in their arguments. We give the list of declared four vectors the name momenta.

To illustrate this scheme assume momenta = {k,p,q} throughout the rest of this section. This list may be enlarged at any time provided symbols that have already been used as Lorentz indices are not included. Let the function g, a symmetric function of two arguments, denote the object that will carry the properties of the metric tensor. What g actually represents will depend on its arguments as follows:

1. g[a,b] represents the metric tensor with Lorentz indices a and b, because a and b are not in the list {k,p,q},

2. g[a,k] represents the four-vector k with Lorentz index a, because only k is in the list {k,p,q}, and

3. g[k,p] represents the scalar product of the four-vectors k and p which are both in {k,p,q}.

Products of gamma matrices are represented by a function d of any non-zero number of arguments. Again, what d represents depends on the arguments. Wherever there appears an argument that is a member of momenta the corresponding matrix is the contraction of the gamma matrix in that position with the four-vector. Otherwise the matrix is simply a gamma matrix with the symbol denoting a Lorentz index. For example, the object d[a,b,k,p,b,d,k,q] denotes

$$\gamma^\alpha \gamma^\beta \not{k} \not{p} \gamma_\beta \gamma^\delta \not{k} \not{q}. \tag{1}$$

3

Free Lorentz indices appearing as arguments of g and d can represent contravariant or covariant indices. In renormalizing quantum field theories we consider only amputated amplitudes. Information about components of four-vectors and matrices is not required. Thus the distinction between contravariant and covariant components is not important here. One simply takes the rank structure of the final expression to be that of the input expression. If one wishes to consider expressions which included external lines then one would need to distinguish between contravariant and covariant indices. This could be done by declaring all covariant indices in a list which one may call covariant. Then symbols which appear in neither of momenta or covariant and were not otherwise defined would implicitly represent contravariant indices.

Traces of products of gamma matrices can be denoted by a function trace whose argument is a linear combination of products of gamma matrices. This function will automatically convert the trace of a linear combination to the linear combination of traces before any traces are evaluated.

Other tensors may be represented by other functions with the rank structure given implicitly by the arguments. For example one may wish to manipulate the Levi-Civita tensor in four dimensions. One would define a function e depending on four arguments. Then, for example e[a,b,u,v] would represent

$$\epsilon^{\alpha\beta\mu\nu} \tag{2}$$

and e[a,b,k,p] would represent

$$\epsilon^{\alpha\beta\mu\nu}k_\mu p_\nu. \tag{3}$$

The mass of a particle is represented by the function M[x] where x is a label for the convenience of the user and is not used by the program. Similarly, other scalars are represented by the function scalar[x]. The left and right helicity projection operators are represented by L and R, respectively. The matrix γ_5 has not been explicitly represented, nor have traces involving γ_5 been implemented. However the definitions could be easily extended if traces involving γ_5 were required (see for example [9]).

Manipulation and simplification of expressions can be performed by machine using pattern recognition, procedural programming, rule based programming and functional programming all of which are supported by Mathematica. One can perform basic manipulations such as

- contracting repeated indices in tensor expressions,

- simplifying an expression of the form $\gamma^\mu \gamma^{\alpha_1} \ldots \gamma^{\alpha_j} \gamma_\mu$

- commuting a gamma matrix through one or more gamma matrices,

- the evaluation of the trace of any number of gamma matrices, and

- evaluation of Feynman integrals

and any other lengthy algebraic manipulation that would be prohibitively tedious to perform manually. Some of the rule definitions which are useful for the manipulations described are illustrated in the next section. Note that the system function Dot has been used in place of d for products of gamma matrices. This function will perform matrix multiplication when explicit matrices in component form are placed in its arguments. Since we require no representation for the gamma matrices and thus only symbols appear as its arguments, the function Dot will simply represent non-commutative multiplication and one can assign rules required to invoke the various properties of Dirac algebra which are required. We have chosen to use Dot for this purpose because a convenient input notation

$$\mathtt{Dot[a1,a2,\ldots]} \equiv \mathtt{a1.a2.\ldots} \tag{4}$$

is available. Since Dot has the attribute OneIdentity the expression Dot[a] is equivalent to a where Head[a] = Symbol. Hence a solitary symbol which is otherwise undefined represents a single gamma matrix.

Together with symbols i for the imaginary unit, n for the number of space-time dimensions, and eps = 4-n, this scheme is sufficient to represent any amputated diagram in a gauge theory. In the next section we will discuss how to manipulate, simplify, and integrate expressions within this scheme.

3   Implementation of Symbolic Algebra

We will begin by discussing expressions representing Lorentz tensors. This will be followed with Dirac algebra and expressions involving both gamma matrices and tensors, and traces of gamma matrices. Finally the evaluation of two-loop Feynman integrals will be discussed. In evaluating Lagrangian counterterms, which in turn give renormalization group functions, we are interested in the pole part of amputated diagrams. This necessitates the evaluation of Laurent expansions and some points relating to this will be discussed briefly.

The metric of n-dimensional Minkowski space is represented by the symmetric function g with two arguments,

$$\mathtt{Attributes[g]} = \{\mathtt{Orderless}\} \tag{5}$$

$$\mathtt{g[x\_,x\_]} := \mathtt{n} \qquad \mathtt{/;\ FreeQ[momenta,x]} \tag{6}$$

$$\mathtt{g[a\_,b\_]\char`\^2} := \mathtt{n}$$
$$\mathtt{/;FreeQ[momenta,a]\ \&\&\ FreeQ[momenta,b]} \tag{7}$$

$$\mathtt{g[a\_,b\_]\char`\^2} := \mathtt{g[b,b]}$$
$$\mathtt{/;FreeQ[momenta,a]\ \&\&\ MemberQ[momenta,b]} \tag{8}$$

Note that the action of the rules is conditional upon whether or not one or both arguments are in the list momenta. In conventional notation the rules stated are

$$g^\alpha_\alpha = n \tag{9}$$

$$g^{\alpha\beta} g_{\alpha\beta} = n \tag{10}$$

$$b^{\alpha} b_{\alpha} = b^2 \tag{11}$$

where b is a vector. Rules can be applied at the discretion of the user by defining new functions which act on tensors. For example the relations

$$g^{\alpha\beta} g_{\beta\lambda} = \delta^{\alpha}_{\lambda} \tag{12}$$

$$b^{\alpha} g_{\alpha\lambda} = b_{\lambda} \tag{13}$$

can be implemented by

$$\mathtt{cc[x\_,a\_,b\_,s\_]} \ := \ \mathtt{x \,/. \, g[a,s] \, g[s,b]} \ \rightarrow \ \mathtt{g[a,b]}$$
$$/; \ \mathtt{FreeQ[momenta,s]} \tag{14}$$

Note that in this notation both (12) and (13) are embodied in one rule. Functions can appear in the definitions of functions. For instance, a function which contracts all occurrences of a repeated index is valuable. Suppose all the arguments of **g** in an expression **x** are placed in a list called **args** except for one repeated index **s**. We have defined a function, **contract[x, args, s]**, which will perform all contractions of the repeated index **s** in **x**. The definition of contract involves two other functions, **cc** which controls the actual replacement that is made, and **ll** which takes a list of symbols and returns a list of all pairs of those symbols. Extending this further, we have defined a function, **contractall[x, repeated, other]**, which takes an expression **x**, a list of all repeated indices placed in the second argument, and a list of all remaining arguments of **g** in **x** in the third argument and performs the contraction of all repeated indices. The definition of this function is in terms of **contract**.

As for **g**, rules are assigned to **Dot** to automatically implement properties of matrix algebra. Note that since **Dot** is a system function it must be unprotected before rules can be added to its definition. The rules we define for **Dot** are applied automatically to any input expression until no further changes occur. These rules correspond to properties such as the distributive law over addition and scalar multiplication. We have chosen to use the symbol **J** to denote the identity matrix. Expressions which represent a combination of tensors and gamma matrices will involve both **g** and **Dot**. In cases where a Lorentz index appears in an argument of both **g** and **Dot** the index may be contracted out of the expression. A function **slash** is defined in the appendix which performs this task.

Identities, such as commutation relations can be applied at the discretion of the user by defining functions which take an expression, search for a specified pattern and replace the pattern by an equivalent expression. For example, the commutation relations

$$\gamma_{\alpha} \gamma_{\beta} = -\gamma_{\beta} \gamma_{\alpha} + 2 g_{\alpha\beta} \tag{15}$$

$$\gamma_{\alpha} \gamma_{\beta} \gamma_{\lambda} = \gamma_{\beta} \gamma_{\lambda} \gamma_{\alpha} + 2\gamma_{\lambda} g_{\alpha\beta} - 2\gamma_{\beta} g_{\alpha\lambda} \tag{16}$$

are applied by the function comm, defined as

$$\mathtt{comm[x\_,a\_,b\_]} \ := \ \mathtt{x \,/. \, a.b} \ \rightarrow \ \mathtt{b.a \ + \ 2J \, g[a,b]} \tag{17}$$

$$\mathtt{comm[x\_,a\_,b\_,c\_]} \ := \ \mathtt{x \,/. \, a.b.c} \ \rightarrow \ \mathtt{b.c.a \ + \ 2c \, g[a,b] \ - \ 2b \, g[a,c]} \tag{18}$$

Which of the two rules is applied depends on how many arguments are passed to comm. Note that the rules are valid whether or not any of a, b and c appear in the list momenta. Further rules have been defined for commuting through more than two gamma matrices, and another function called rcomm for commuting in the reverse order. Rules which apply identities for expressions of the form

$$\gamma_\mu \gamma_{\alpha_1} \cdots \gamma_{\alpha_j} \gamma^\mu \tag{19}$$

$$\rlap{/}{k} \gamma_{\alpha_1} \cdots \gamma_{\alpha_j} \rlap{/}{k} \ , \ j = 1, 2, 3, \ldots \tag{20}$$

have been similarly defined. For instance

$$
\begin{aligned}
\text{con}[\text{x\_}, \text{a\_}] \ := \ &\text{If}\,\big[\,\text{MemberQ}[\text{momenta}, \text{a}]\,, \\
&\qquad \text{x}\,/.\,\text{a.a} \rightarrow \text{g}[\text{a}, \text{a}]\,\text{J}, \\
&\qquad \text{x}\,/.\,\text{a.a} \rightarrow \text{n}\,\text{J} \\
&\,\big]
\end{aligned} \tag{21}
$$

$$
\begin{aligned}
\text{con}[\text{x\_}, \text{a\_}, \text{b\_}] \ := \ &\text{If}\,\big[\text{MemberQ}[\text{momenta}, \text{a}]\,, \\
&\qquad \text{x}\,/.\,\text{a.b.a} \rightarrow -\text{b}\,\text{g}[\text{a}, \text{a}]\,+\,2\text{a}\,\text{g}[\text{a}, \text{b}], \\
&\qquad \text{x}\,/.\,\text{a.b.a} \rightarrow (2-\text{n})\,\text{b} \\
&\,\big]
\end{aligned} \tag{22}
$$

The answer is dependent on whether or not the repeated symbol a appears in the list momenta. Rules for $j > 1$ can be included as needed.

To simplify expressions involving helicity projection operators, the functions movel and mover have been defined. They, respectively, move the helicity projection operators to the left or to the right in each term in an expression.

The trace of an arbitrary linear combination of products of gamma matrices can be evaluated using the linearity property of traces and the recursive relation

$$Tr(\gamma^{\alpha_1} \gamma^{\alpha_2}) \ = \ 4g^{\alpha_1 \alpha_2} \tag{23}$$

$$Tr(\gamma^{\alpha_1} \ldots \gamma^{\alpha_{2j}}) \ = \ \sum_{l=o}^{2j-2} (-1)^l g^{\alpha_1 \alpha_{2j-l}} Tr(\gamma^{\alpha_2} \ldots \hat{\gamma}^{\alpha_{2j-l}} \ldots \gamma^{\alpha_{2j}}) \tag{24}$$

$$Tr(\gamma^{\alpha_1} \ldots \gamma^{\alpha_{2j+1}}) \ = \ 0 \tag{25}$$

where the hat over the gamma matrix denotes its absence [9]. The function trace will firstly convert the trace of a linear combination into a linear combination of traces, and then applies a function tr which makes use of the recursion relation (24) to evaluate the traces. Note that tr first checks that the length of the product is even and returns zero if it is not. The case of a product of two gamma matrices acts as an initial condition.

The scheme described so far can be used to represent any Feynman integral which can arise in a gauge theory. A means of evaluating the integrals would complete a scheme

for the evaluation of amputated Feynman diagrams. We will present a simple procedure which can be adapted to all cases.

We begin by considering integrals of the form

$$\mu^\lambda \int \frac{d^n q}{(2\pi)^n} \frac{q_\alpha q_\beta \cdots}{(q^2)^j((q \pm p)^2)^l} \tag{26}$$

and

$$\mu^\lambda \int \frac{d^n q}{(2\pi)^n} \frac{\not{q} q_\alpha q_\beta \cdots}{(q^2)^j((q \pm p)^2)^l} \tag{27}$$

where μ is an arbitrary mass scale, q is the integration variable and the rank of the tensor in the numerator is usually less than or equal to 3. The factor μ^λ is introduced to ensure that the integral is dimensionless for any n (hence λ depends on n). This is sufficient for most two loop calculations but if rules for the integration of higher rank expressions are needed, then extra rules can be included.

The main consideration in defining an integration function is to be certain that there is no residual dependence on the integration variable in the final expression. For example, a function could be defined which searches for the pattern

$$\frac{1}{g[q,q]\,g[q+p,q+p]} \tag{28}$$

and replaces it using a replacement rule. If that function were then applied to a pattern like

$$\frac{T[q]}{g[q,q]\,g[q+p,q+p]} \tag{29}$$

the integration function would treat this as a product of two factors

$$T[q] \frac{1}{g[q,q]\,g[q+p,q+p]} \tag{30}$$

and replace only the second factor, leaving T[q] in the resulting expression. Hence it is important to check the final expression for the presence of the integration variable.

In the integrals (26) and (27) there are other considerations. The rank, the symbols used for indices, the integration variable, the parameter p , and the powers j and l should all be variables which can be passed to the integrating function. We have defined functions inttensor and intgamma which will return the integrals of (26) and (27) respectively. The arguments to be passed to inttensor and intgamma are

$$\text{inttensor}[x, q, p, a, b, \ldots, j, l] \tag{31}$$

and

$$\text{intgamma}[x, q, p, a, b, \ldots, j, l], \tag{32}$$

where x is the expression to be integrated, q is the integration variable, p is the four-momentum parameter, a,b,... are Lorentz indices, and j and l are the powers in

8

the denominator. Again the definitions of `inttensor` and `intgamma` consist of a limited series of rules. Which of the rules is applied depends on the number of arguments passed to `inttensor` and `intgamma`. If the rules do not cover cases of high rank that may be needed then new rules may be included as needed. Rank three integrals are usually sufficient for most two-loop calculations. Integrals involving non-zero masses have not been implemented, though a function for these could be defined.

Dimensionally regularised integrals can be expanded as a Laurent series in $\epsilon = 4 - n$. In general an integral over l four vectors will have a leading pole of order ϵ^{-l}. To evaluate counterterms only pole parts of the Laurent expansions of integrals need be evaluated. A table of pole parts of integrals can be prepared with the help of `inttensor` and `intgamma`. This table can then be used to evaluate the pole parts of diagrams and hence the counterterms. The integrals required for this can be classified according to the form of the denominators of the Feynman integrals. For example, to calculate the two-loop boson self energy in a $SU(m)$ gauge theory with massless fermions there is just one class of integrals as the Feynman integral is always of the form

$$\mu^\lambda \int_{q,k} \frac{T_{\mu\nu}(q,k,p)}{q^2(q+p)^2 k^2(k+p)^2(k-q)^2} \tag{33}$$

where p is the external momentum, and $T_{\mu\nu}(q,k,p)$ is a rank-2, dimension 4 tensor constructed from q, k, and p. There are 75 distinct possibilities for $T_{\mu\nu}$ but fortunately many of the corresponding integrals are related, and the number of independent integrals is reduced to about 10. If we are interested only in the pole parts then we can express any integral of the form (33) in terms of the pole parts of just four integrals. We choose the four integrals where $T_{\mu\nu}(q,k,p)$ is either

$$t_{\mu\nu}(p)(q+p)^2 \tag{34}$$

or

$$t_{\mu\nu}(p)(k-q)^2 \tag{35}$$

and $t_{\mu\nu}(p)$ is either $p_\mu p_\nu$ or $g_{\mu\nu}p^2$.

In the integrals discussed we have represented the Euler gamma function by `gam[x]`. For the evaluation of pole parts of one-particle irreducible amplitudes one must be able to perform the Laurent expansions. This can be done by either replacing `gam` with the system defined version of the Euler gamma function and using such functions as `Series`, or by defining a sufficient set of rules for `gam` to evaluate the expansion.

4   Examples

As an illustration of the application of this work to some specific examples, we will briefly discuss the evaluation of the integral

$$\mu^{2(4-n)} \int \frac{d^n q}{(2\pi)^n} \frac{d^n k}{(2\pi)^n} \frac{1}{k^2(k+p)^2 q^2(q-k)^2} \tag{36}$$

the evaluation of the diagram shown in figure **??**, and the simplification of the diagram in figure **??**. Diagram **??** is a contribution to the gauge boson self energy in the presence of massless fermions in some representation R of the gauge group. The diagram **??** represents a contribution to the quark-photon vertex in the Weinberg-Salam model.

Before evaluating the integral (36) we must first declare the list of four-vector symbols as

$$\texttt{momenta} = \{\mathbf{k}, \mathbf{p}, \mathbf{q}, \mathbf{k} + \mathbf{p}, \mathbf{q} + \mathbf{p}, \mathbf{k} - \mathbf{q}, \mathbf{q} - \mathbf{k}\} \tag{37}$$

Then we specify the integrand by

$$\mathbf{x} = \frac{1}{\mathbf{g}[\mathbf{k}, \mathbf{k}] \, \mathbf{g}[\mathbf{k} + \mathbf{p}, \mathbf{k} + \mathbf{p}] \, \mathbf{g}[\mathbf{q}, \mathbf{q}] \, \mathbf{g}[\mathbf{q} - \mathbf{k}, \mathbf{q} - \mathbf{k}]} \tag{38}$$

The integral over **q** is performed using inttensor,

$$\mathbf{x} = \texttt{inttensor}[\mathbf{x}, \mathbf{q}, \mathbf{k}, 1, 1] \tag{39}$$

which results in an expression depending on the integration variable **k**. The integrand of this expression is proportional to

$$\frac{1}{(\mathbf{g}[\mathbf{k}, \mathbf{k}])^{1 + \frac{\texttt{eps}}{2}} \, \mathbf{g}[\mathbf{k} + \mathbf{p}, \mathbf{k} + \mathbf{p}]} \tag{40}$$

and can also integrated by using **inttensor** as follows.

$$\mathbf{x} = \texttt{inttensor}[\mathbf{x}, \mathbf{k}, \mathbf{p}, 1 + \texttt{eps}/2, 1] \tag{41}$$

The resulting expression is

$$\frac{-\left(2^{2\,\texttt{eps}}\, \texttt{mu}^{2\,\texttt{eps}}\, \texttt{pi}^{-4+\texttt{eps}}\, \texttt{gam}[1 - \texttt{eps}]\, \texttt{gam}[1 - \frac{\texttt{eps}}{2}]\, \texttt{gam}[\frac{\texttt{eps}}{2}]\, \texttt{gam}[\texttt{eps}]\right)}{256\,(-1)^{\texttt{eps}}\, \mathbf{g}[\mathbf{p}, \mathbf{p}]^{\texttt{eps}}\, \texttt{gam}[2 - \frac{3\,\texttt{eps}}{2}]\, \texttt{gam}[2 - \texttt{eps}]\, \texttt{gam}[1 + \frac{\texttt{eps}}{2}]} \tag{42}$$

where **mu** is the introduced mass scale μ, and this can be Laurent expanded to order ϵ^{-1}. We obtain

$$\frac{-2 + 2\,\texttt{elog} - 5\,\texttt{eps}}{256\,\texttt{eps}^2\,\texttt{pi}^4} + \texttt{O}(\texttt{eps})^0 \tag{43}$$

where

$$\texttt{elog} = \texttt{egam} - \log\left(\frac{4\,\texttt{pi}\,\texttt{mu}^2}{-\mathbf{g}[\mathbf{p}, \mathbf{p}]}\right) \tag{44}$$

and **egam = 0.577...** is Euler's constant. This gives the result for two of the four integrals required to determine the pole part of (33) and the remaining integrals can be evaluated similarly. Then to integrate the expression (33) we first replace the integrand with an expression that has the same pole part and then replace the integrals with the Laurent expansions. The functions we have defined to do this are **samepole** and **poleform** respectively.

To evaluate the diagram of figure **??** we declare the momenta list to be (37), and define some initial expressions. In the notation of our scheme they are the coefficient of the integral, the numerator and denominator of the integrand. The coefficient is

$$\texttt{coefficient} = \frac{\texttt{i}}{2}\,\texttt{g}^4\,\texttt{C[G]}\,\texttt{T[R]}\,\texttt{delta[a,b]} \tag{45}$$

where $\texttt{a}$ and $\texttt{b}$ are gauge group indices, $\texttt{delta}$ is the delta symbol and $\texttt{C[G]}$ and $\texttt{T[R]}$ are gauge group factors. The numerator of the integrand has two factors

$$\texttt{y} = (\texttt{g[p,a]} - \texttt{g[k,a]})\,\texttt{g[u,l]} + (2\texttt{g[k,u]} + \texttt{g[p,u]})\,\texttt{g[l,a]}-$$
$$(2\texttt{g[p,l]} + \texttt{g[k,l]})\,\texttt{g[a,u]} \tag{46}$$

and

$$\texttt{z} = \texttt{trace[a.(p + q).v.q.l.(k - q)]} \tag{47}$$

and the denominator is

$$\texttt{denominator} = \texttt{g[q,q]}\,\texttt{g[q + p,q + p]}\,\texttt{g[k,k]}\,\texttt{g[k + p,k + p]}\,\texttt{g[k - q,k - q]} \tag{48}$$

In the numerator there are two repeated indices. We set $\texttt{repeated} = \{\texttt{a,l}\}, \texttt{other} = \{\texttt{u,v,k,p,q}\}$ and $\texttt{x} = \texttt{Expand[y\,z]}$ and remove the repeated indices from $\texttt{x}$ using $\texttt{contractall}$,

$$\texttt{x} = \texttt{contractall[x,repeated,other]} \tag{49}$$

All scalar products are then eliminated using $\texttt{preps}$ and $\texttt{prepd}$ which make use of identities such as

$$k.p = \frac{1}{2}((k + p)^2 - k^2 - p^2). \tag{50}$$

The resulting expression for $\texttt{x}$ consists of 69 terms. We then replace $\texttt{x}$ with an expression with the same pole part using $\texttt{samepole}$. The result is divided by $\texttt{denominator}$ and the Laurent expansion is evaluated by applying $\texttt{poleform}$. Simplifying the result and multiplying by $\texttt{coefficient}$ gives the final result for the diagram

$$-(\texttt{g}^4\,\texttt{i}\,\texttt{C[G]}\,\texttt{T[R]}\,\texttt{TR[J]}\,\texttt{delta[a,b]}$$
$$((-48 - 76\,\texttt{eps} + 48\,\texttt{elog}\,\texttt{eps})\,\texttt{g[p,u]}\,\texttt{g[p,v]} +$$
$$(12 - 59\,\texttt{eps} - 12\,\texttt{elog}\,\texttt{eps})\,\texttt{g[p,p]}\,\texttt{g[u,v]}))/(55296\,\texttt{eps}^2\,\texttt{pi}^4) \tag{51}$$

where $\texttt{TR[J]}$ is the trace of the spin identity matrix.

We have used our scheme to evaluate the gauge boson, fermion and ghost anomalous dimensions to two loops. We have also calculated the coupling constant renormalization by evaluating the gauge boson - ghost vertex, and evaluated the Callan-Symanzik beta function. Our results are in complete agreement with [14] for these calculations.

Now consider the diagram in figure **??**, which is a two loop diagram contributing to the quark-photon vertex. The one loop, flavour-changing quark self energy, with momentum p can be written as

$$A\,\not{p}L + B\,\not{p}R + C\,L + D\,R \tag{52}$$

11

where A, B, C and D are scalars depending upon p^2 and the quark masses. After defining

$$\texttt{momenta} = \{\texttt{k},\texttt{p}\} \tag{53}$$

the numerator of the diagram in figure **??** is proportional, in our notation, to

$$
\begin{aligned}
\texttt{x} \;=\;\; &\texttt{alpha.L.(p + k + M[s]).(scalar[A](p + k).L + scalar[B](p + k).R}\\
&\texttt{+scalar[C]L + scalar[D]R).(p + k + M[d]).mu.(p - k + M[d]).}\\
&\texttt{alpha.L}
\end{aligned}\tag{54}
$$

The masses are implicitly multiplied by the identity matrix. To simplify this we first apply

$$\texttt{x = movel[x]} \tag{55}$$

Then repeated application of the function $\texttt{con}$ will do the contractions over γ_α and contract the pairs of $\not{k}$'s and $\not{p}$'s. For example, the following does all possible contractions over $\not{k}$ and $\not{p}$.

$$
\begin{aligned}
\texttt{x} \;=\;\; &\texttt{con[x,p]}\,;\,\texttt{x = con[x,k]}\,;\,\texttt{x = con[x,p,mu]}\,;\\
&\texttt{x = con[x,k,mu]}\,;\,\texttt{x = con[x,p,k]}\,;\,\texttt{x = con[x,k,p]}\,;\\
&\texttt{x = con[x,p,mu]}\,;\,\texttt{x = con[x,k,mu]}
\end{aligned}\tag{56}
$$

A further seven applications of $\texttt{con}$ are necessary to do all the contractions over γ_α. In the resulting expression, repeated use of $\texttt{comm}$

$$
\begin{aligned}
\texttt{x} \;=\;\; &\texttt{comm[x,p,k]}\,;\,\texttt{x = comm[x,p,mu]}\,;\,\texttt{x = comm[x,k,mu]}\,;\\
&\texttt{x = comm[x,p,k]}
\end{aligned}\tag{57}
$$

followed by collecting with respect to $\texttt{Dot}$, produces a final expression of four terms, proportional to $R\gamma_\mu\not{k}\not{p}$, $R\gamma_\mu$, $R\not{k}$ and $R\not{p}$. The full expression has seventy-six terms when expanded and is too long to reproduce here.

5 Conclusion

We have presented a scheme for representing amputated Feynman diagrams beyond tree level in terms of Mathematica expressions and have shown how the algebra involved in simplifying these expressions and evaluating the integrals can be automated using the symbolic algebra capacity of Mathematica. This scheme is based upon the specification of a sufficiently convenient notation for Lorentz tensors and Dirac algebra. An automatic procedure for the evaluation of pole parts for dimensionally regularised massless Feynman integrals has been used to evaluate two loop counterterms in a non-abelian gauge theory with fermions and for the simplification of amputated amplitudes in the Weinberg-Salam model. The methods used in this paper can be readily extended to handle integrals of higher loop order, and integrals involving masses. Our approach succeeds in eliminating all tedious hand calculations. The time required to perform calculations automatically is generally small compared to the time required to prepare input and organise a calculational sequence.

6 Acknowledgements

We would like to thank A. A. Rawlinson and A. C. Kallionatis for their useful suggestions and assistance.

7 Appendix

This appendix contains the usage messages for functions described in this article.

The following usage messages are for functions which simplify products of Dirac gamma matrices.

```
comm::usage = "comm[x, a, b1, b2, ...]  commutes a to the
right through b1.b2 ...wherever the pattern a.b1.b2 ...
appears in x (up to 5 b's programmed)."

rcomm::usage = "rcomm[x, b1, b2, ..., a] commutes a to
the left through b1.b2 ...wherever the pattern b1.b2 ....a
appears in x (up to 5 b's programmed)."

con::usage = "con[x, a, b1, b2, ...]  evalates a.b1.b2 ....a
by contracting out the a's.  Replacement made wherever the
pattern appears in x (Up to 6 b's programmed)."

trace::usage = "trace[x] evaluates the trace of a linear
combination of products of gamma matrices that does not
include gamma_5."

movel::usage = "movel[x] commutes all helicity projection
operators to the left most position in each term in the
expression x."

mover::usage = "mover[x] commutes all helicity projection
operators to the right most position in each term in the
expression x."
```

The following usage messages are for functions which simplify Lorentz tensors.

```
contract::usage = "contract[x, other, s] will eliminate
repeated index s from x.  A list other, of all other
symbols appearing in the arguments of g's must be passed
to the function."

contractall::usage = "Given a list of repeated indices,
repeated, and a list, other, of any indices appearing in
the arguments of g's that are not in repeated,
contractall[x, repeated,other] will eliminate all
of the repeated indices from x."
```

```
slash::usage = "slash[x, a, b] finds all patterns of the
form c1.c2.  ....ci.a.d1.d2.  ....dj*g[a,b] in x and
replaces them with c1.c2.  ....ci.b.d1.d2.  ....dj
where either or both of i and j may be zero."

preps::usage = "preps[x, k, p] replaces g[k, p] by
(g[k+p, k+p] - g[k, k] - g[p, p])/2 everywhere in x."

prepd::usage = "prepd[x, k, p] replaces g[k, p] by
(g[p, p] + g[k, k] -g[k-p, k-p])/2 everywhere in x."
```

The following usage messages are for functions which perform Feynman integration.

```
inttensor::usage = "inttensor[x, q, p, a, b, ..., j, l]
integrates tensors (g[q, a] g[q, b] ...)/(g[q, q]^j g[q ±
p, q ± p]^l) of low rank, where the integration variable
q, external momentum p, powers j and l, and indices
a,b ...are specified in the arguments of inttensor."

intgamma::usage = "intgamma[x, q, p, a, b, ..., j, l]
integrates matrices (q g[q, a]  g[q, b] ...)/(g[q, q]^j g[q
± p, q ± p]^l) of low rank, where the integration
variable q, external momentum p, powers j and l, and
indices a,b ...are specified in the arguments of intgamma."

samepole::usage = "samepole[x, p] replaces x, the numerator
of a Feynman integral for a gauge boson self energy diagram
with a simpler expression that has the same pole
part (p is the external momentum)."

poleform::usage = "poleform[x, p] evaluates the pole part
of x, a Feynman integral for a gauge boson self energy
diagram, after samepole has been applied to the numerator."
```

References

[1] O. V. Tarasov, A. A. Vladimirov and A. Yu. Zharkov, Phys. Lett. **93B** (1980) 429.

[2] K. G. Chetyrkin, A. L Kataev and F. V. Tkachov, Phys. Lett. **99B** (1981) 147; K. G. Chetyrkin, S. G. Gorishny, S. A. Larin and F. V. Tkachov, Phys. Lett. **132B** (1983) 351; D. I. Kazakov, Phys. Lett. **133B** (1983) 406.

[3] H. Strubbe, Comput. Phys. Commun. **8** (1974) 1; V. P. Gerdt, O. V. Tarasov and D. V. Shirkov, Usp. Fiz. Nauk **130** (1980) 113 [Sov. Phys. Usp. **23** (1980) 59].

[4] J. Küblbeck, M. Böhm and A. Denner, Comput. Phys. Commun. **60** (1990) 165; R. Mortig, M. Böhm and A. Denner, Comput. Phys. Commun. **64** (1991) 345.

[5] A. Hsieh and E. Yehudai, Comput. Phys. **6**, (1992) 253.

[6] J. Beringer in *La Londe-les-Maures* 1992, Proceedings, New Computing Techniques in Physics Research II.

[7] J. Fleischer and O. V. Tarasov, Comput. Phys. Commun. **71** (1992) 193.

[8] M. Jamin and E. Lantenbacher, Comput. Phys. Commun. **74** (1993) 265.

[9] T. H. West, Comput. Phys. Commun. **77** (1993) 286.

[10] S. Wolfram, *Mathematica*: A System for Doing Mathematics by Computer (Addison-Wesley, Redwood City, California, 1988).

[11] J. A. M. Vermaseren, FORM User's guide, Nikhef, Amsterdam (1990).

[12] S. A. Larin and J. A. M. Vermaseren, Z. Phys. **C57** (1993) 93.

[13] G. t'Hooft and M. Veltman, Nucl. Phys. **B44** (1972) 189.

[14] E. Sh. Egoryan and O. V. Tarasov, Teor. Mat Fiz. **41** (1979) 26; W. E. Caswell, Phys. Rev. Lett. **33** (1974) 244; D. R. T. Jones, Nucl. Phys. **B75** (1974) 53.